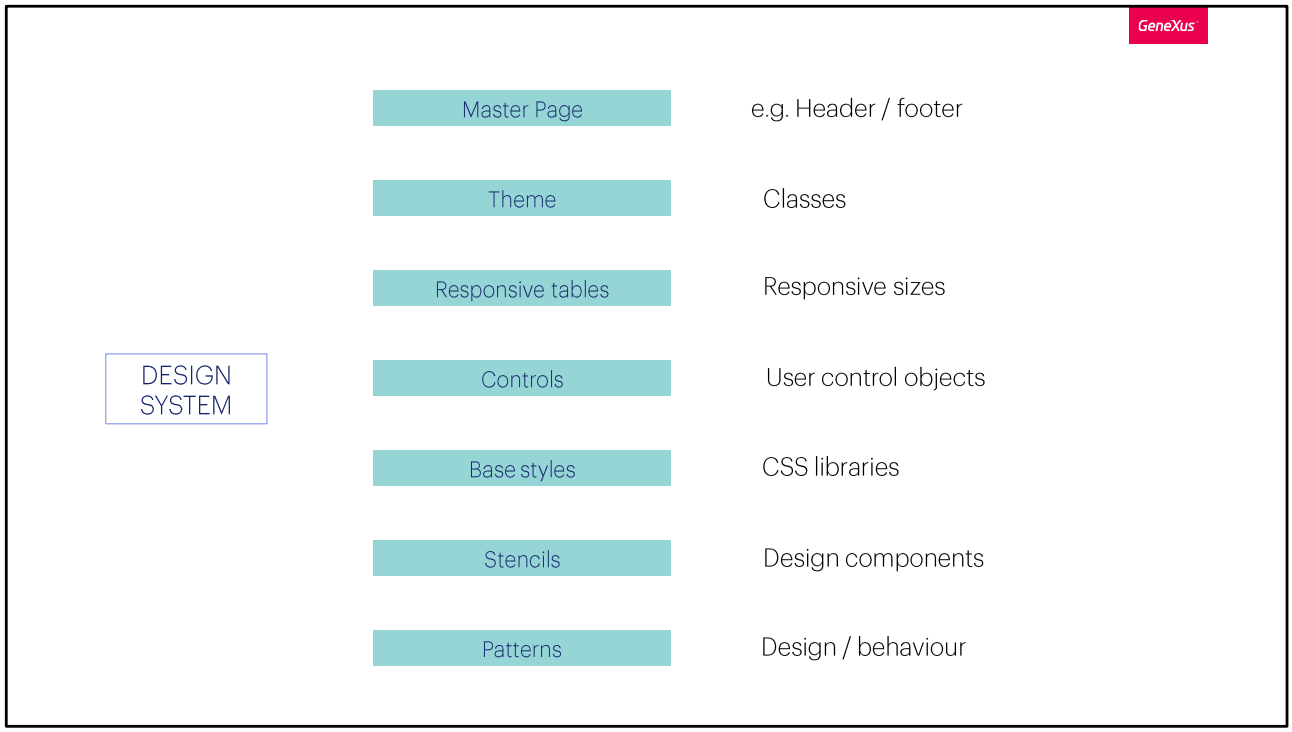
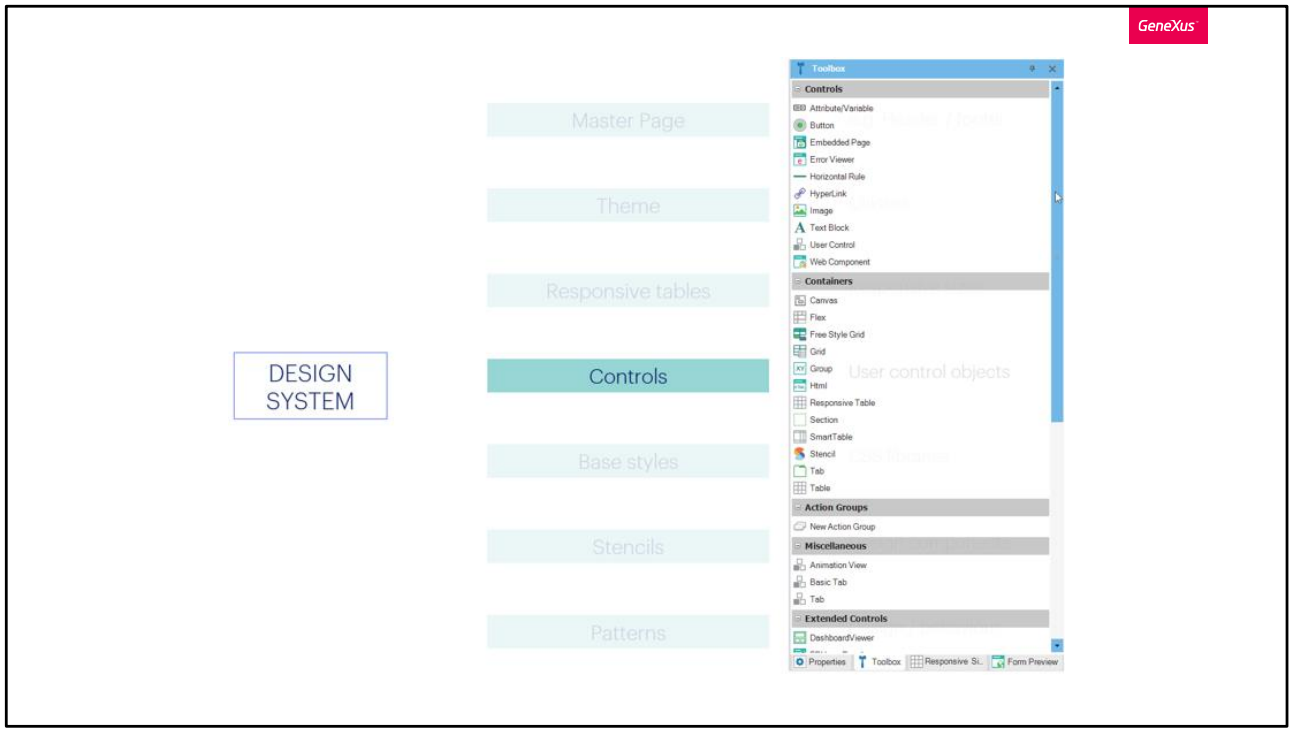
User Control Object

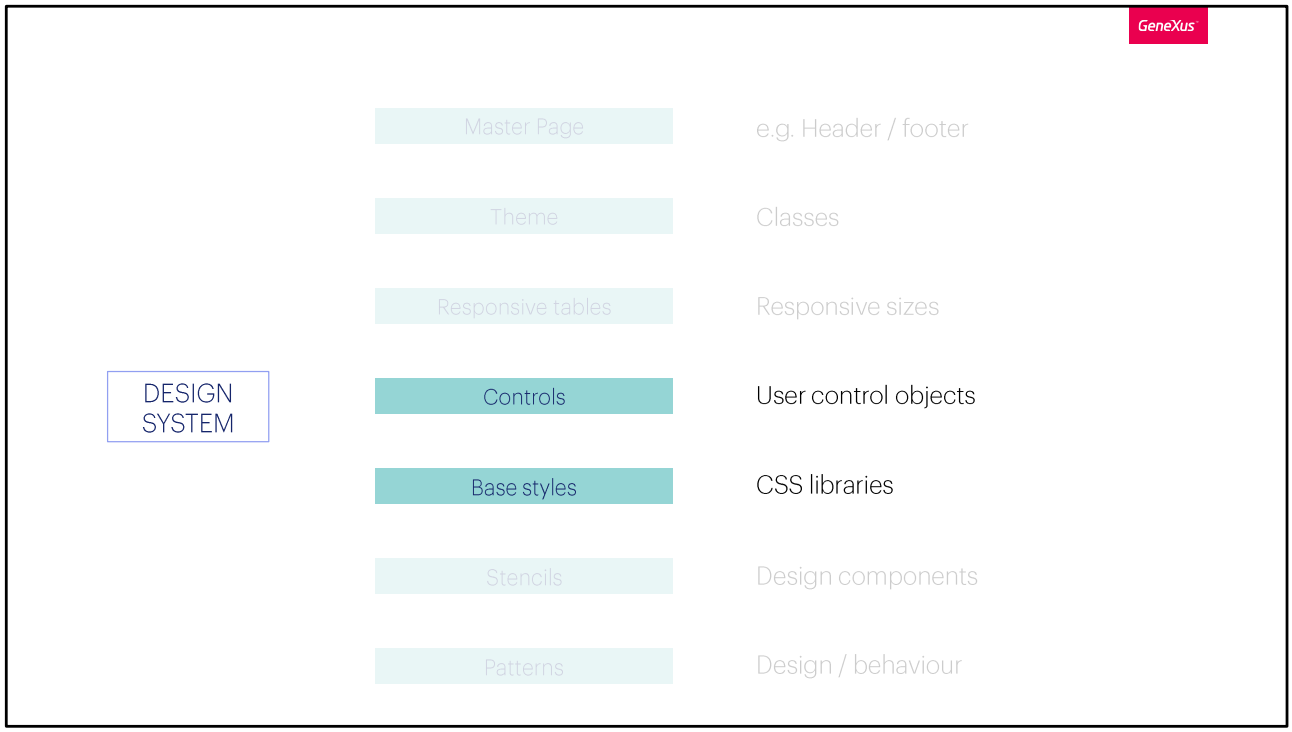
GeneXus™



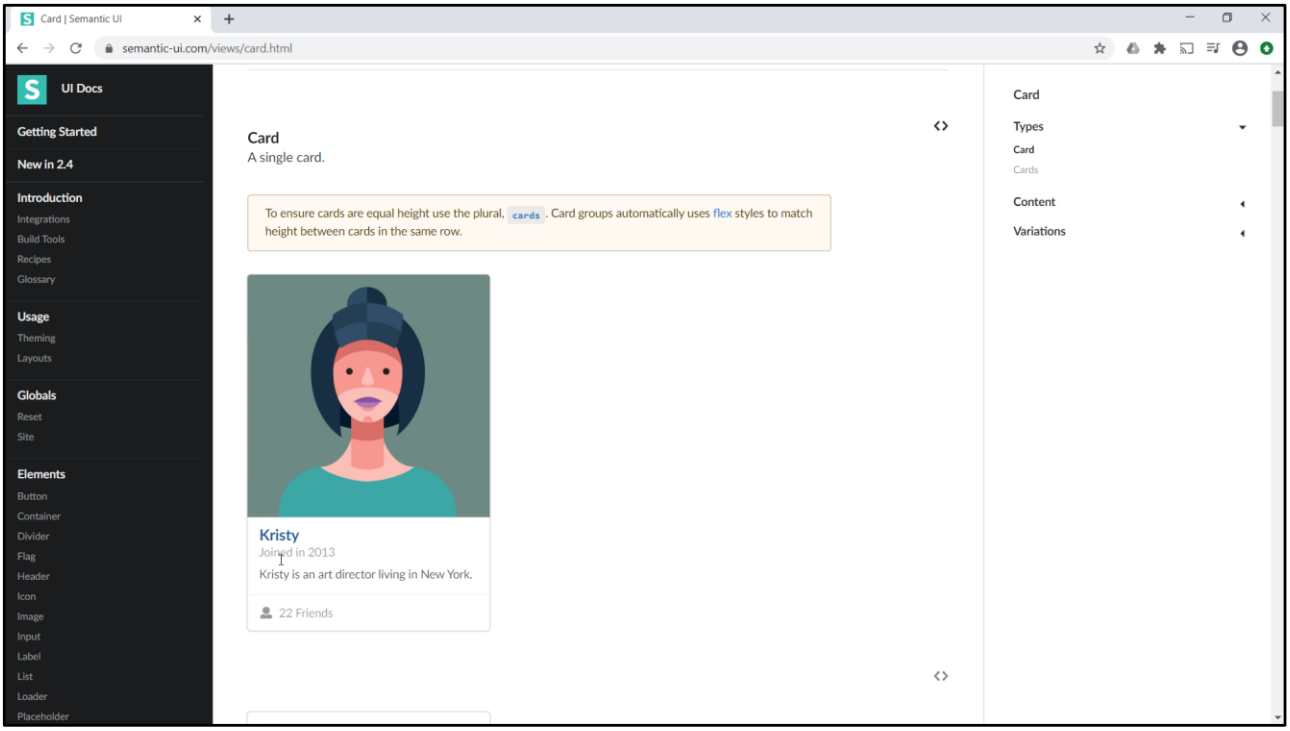
In the initial GeneXus course, we briefly covered each of the GeneXus elements involved in the implementation of a Design System for the application.



Among them, we had said that we could not only use in our forms the controls that come in the GeneXus toolbox by default...

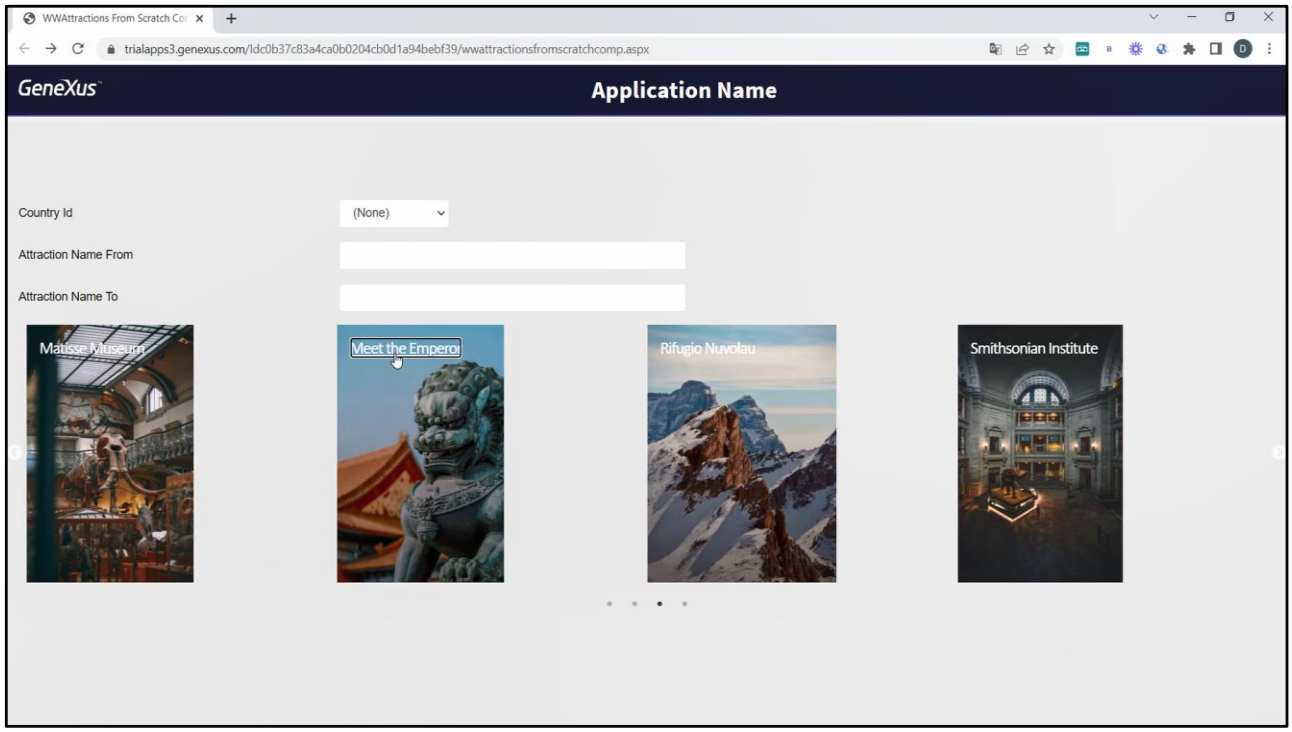


...but we could also define user controls, which can be copied from platforms that offer those controls together with CSS libraries (which are, like our themes, the ones that specify their style –the style of those controls).

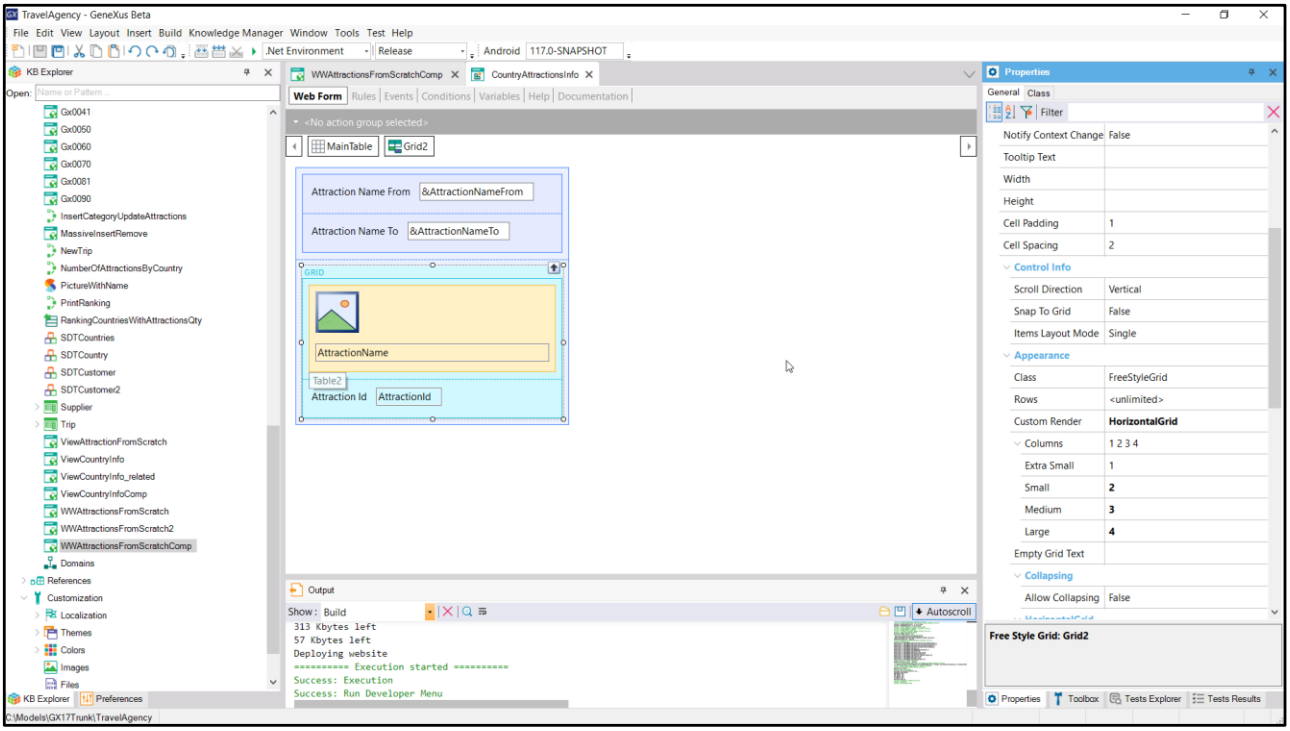


For example, this one, Semantic UI.

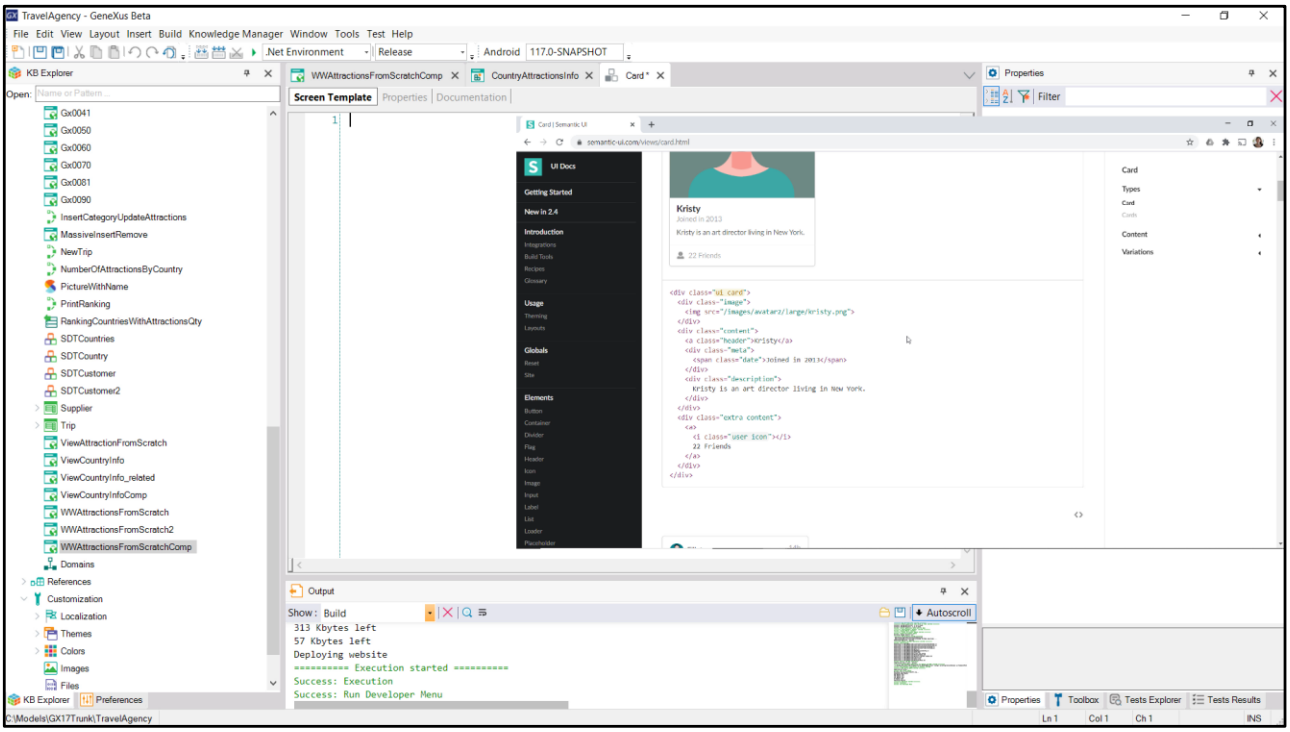
We may want to use a Card control such as this one, for example...



...to show the tourist attractions instead of the canvas table we had placed on the horizontal grid, which we had just started to design.

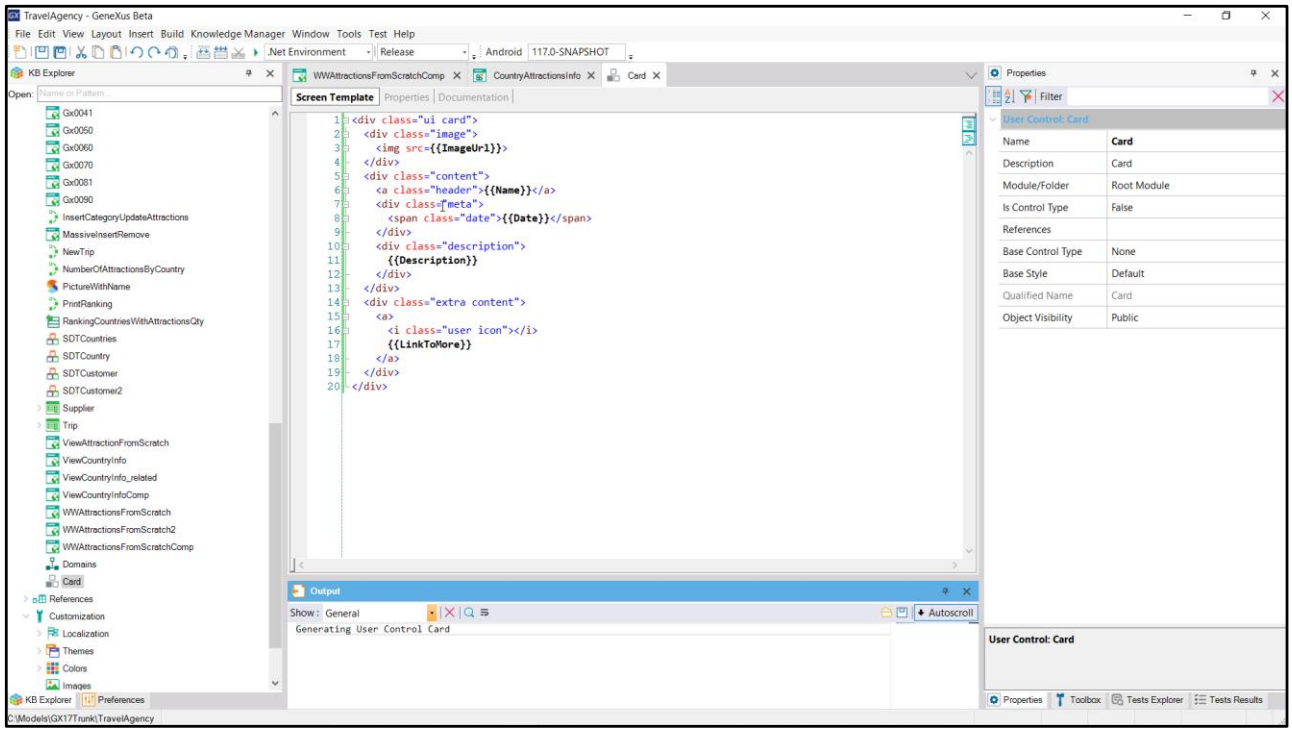


There we can see the grid and here is the canvas table.



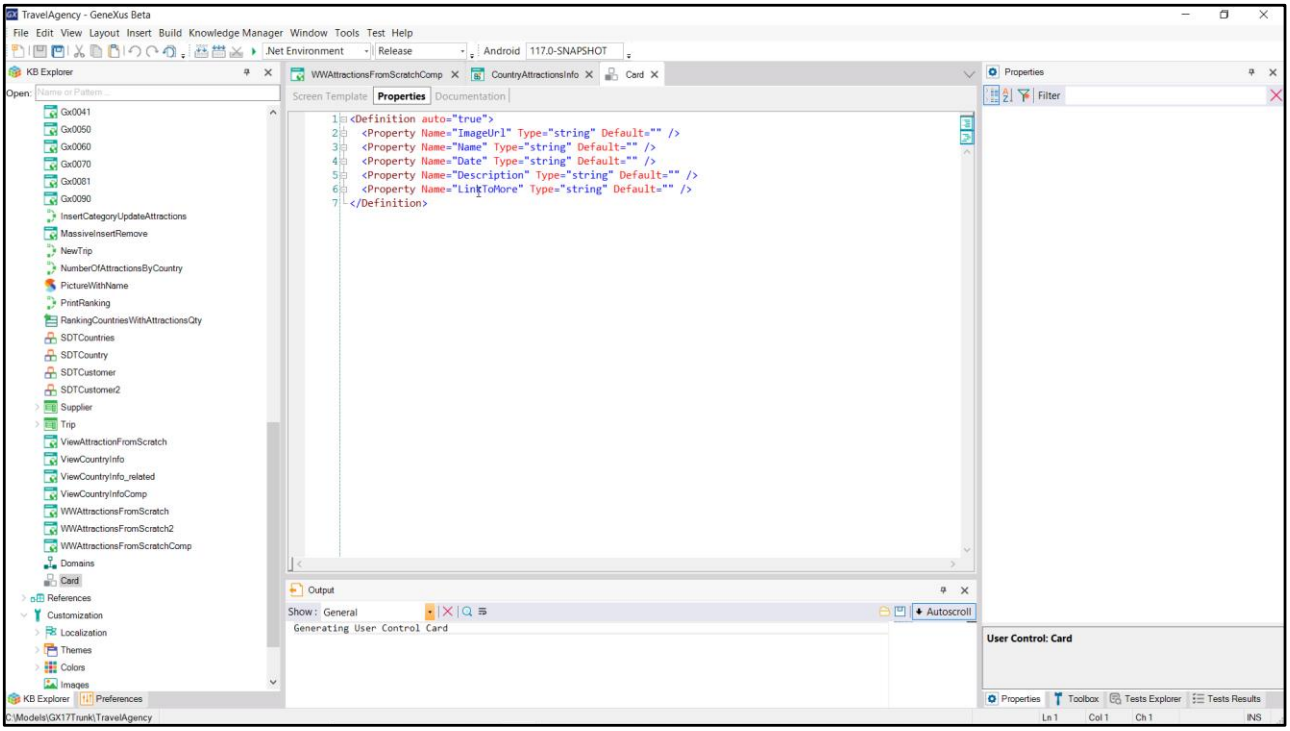
To do this, it is enough to create a User control object (we can call it the same)...

...copy and paste the HTML that Semantic UI offers...

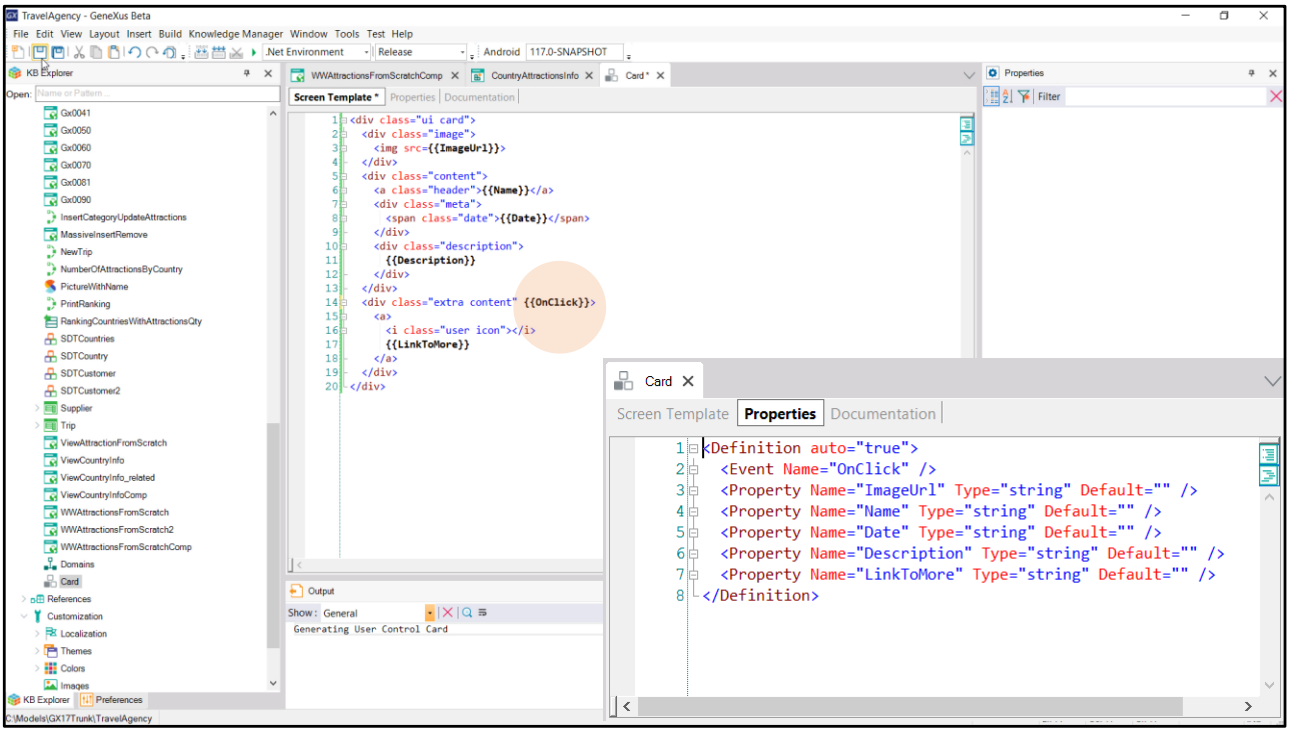


...change this fixed data to something like names of SDT elements (to be able to use them dynamically, i.e., to dynamically load that control with data that we will be able to specify, for example, from the database).

With this syntax, which is known as moustache syntax, we are then changing fixed data into variable data.



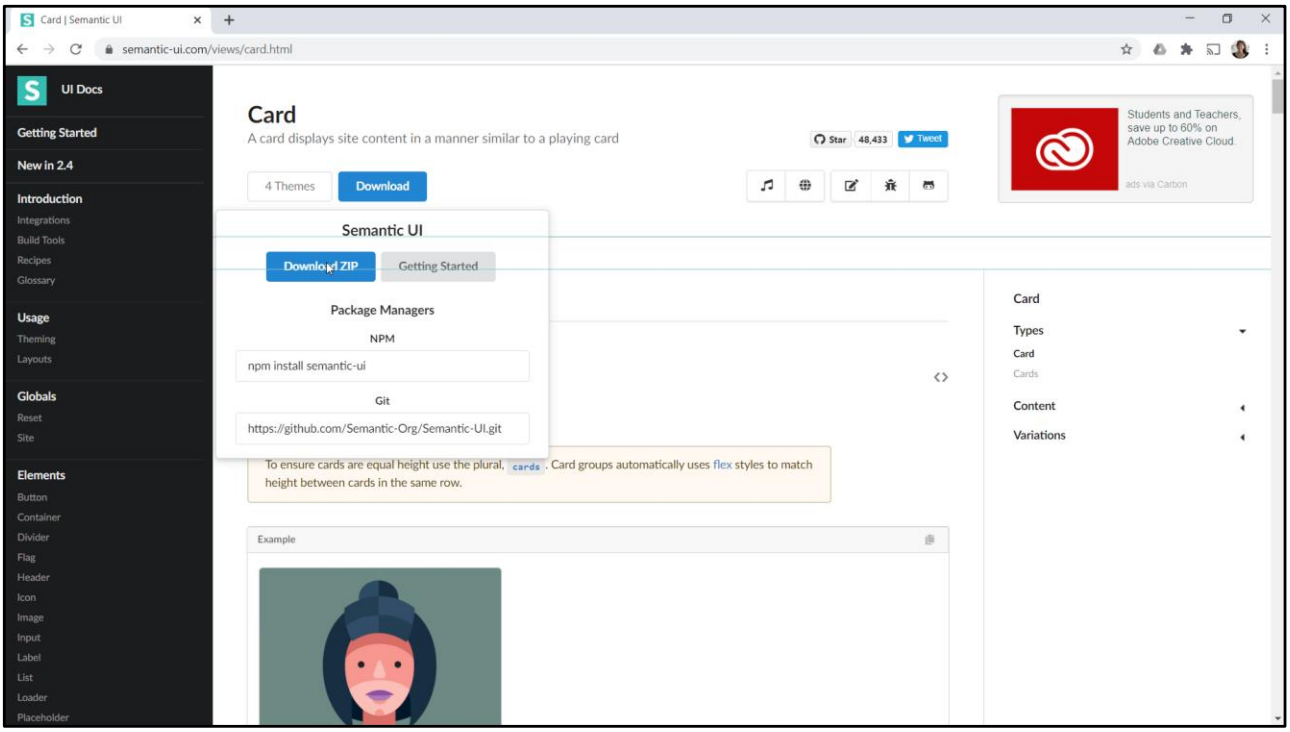
If we go now to the Properties, we see that they are appearing there.



We will also add an event, which is the event that occurs when you click on this HTML component.

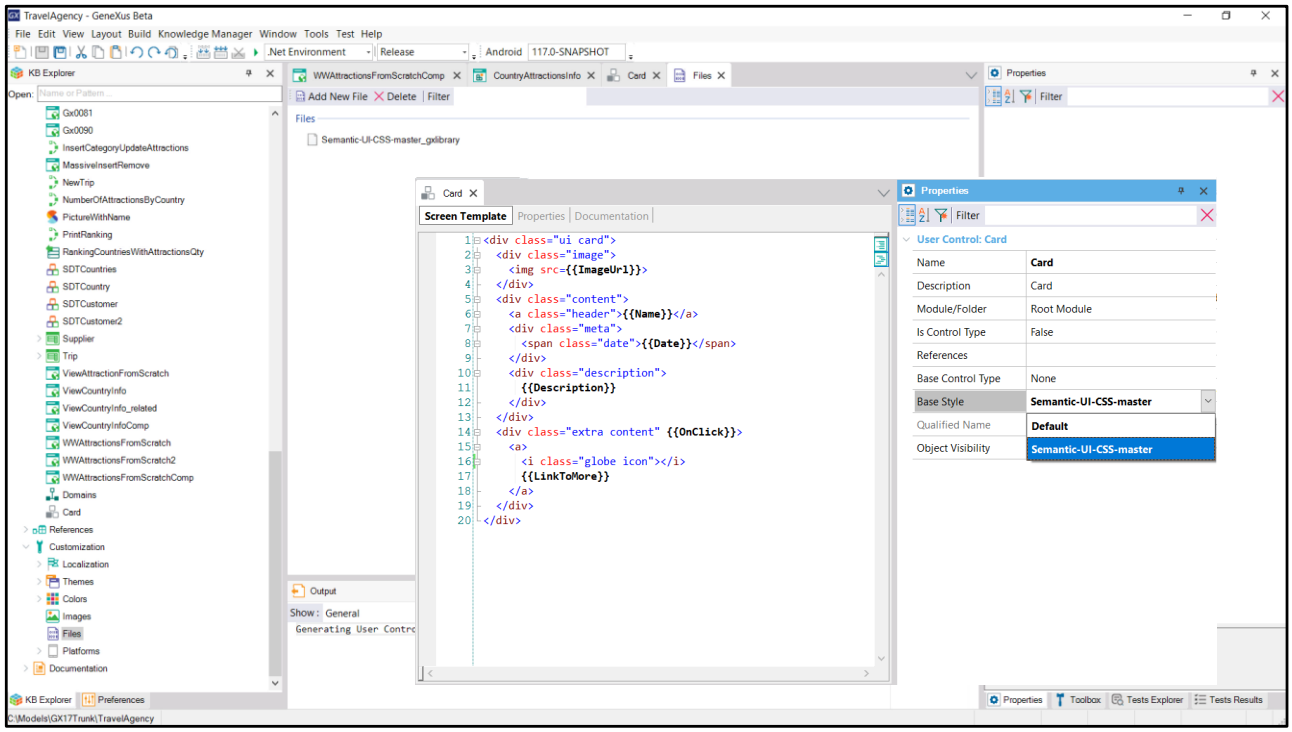
And we see that it is displayed here.

In this way, we will be able to program the click on that element at runtime.



What we are going to do now is download the CSS, which is the one that really implements the design of all these HTML tags.

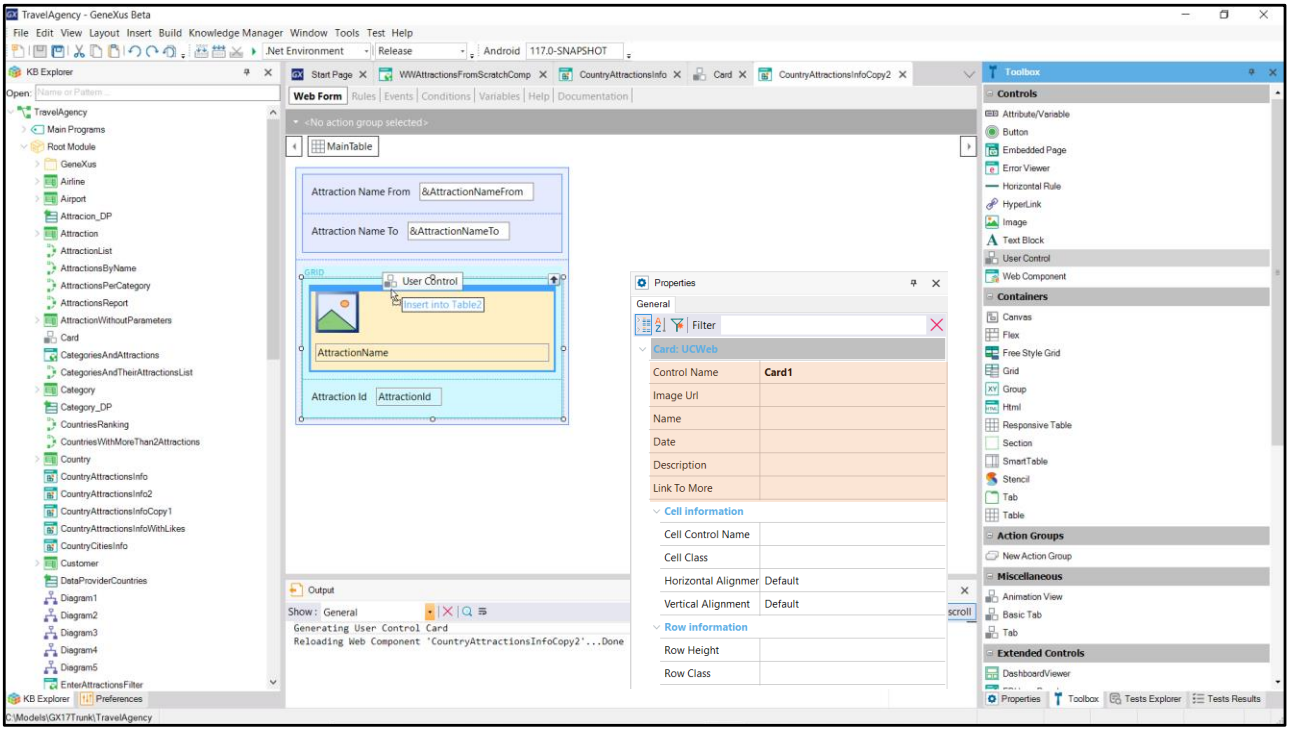
Here we have it. We download it as a zip file. We will change the extension to gxlibrary, which is the extension required in GeneXus to interpret this CSS.



Now we are going to insert this CSS as a file.

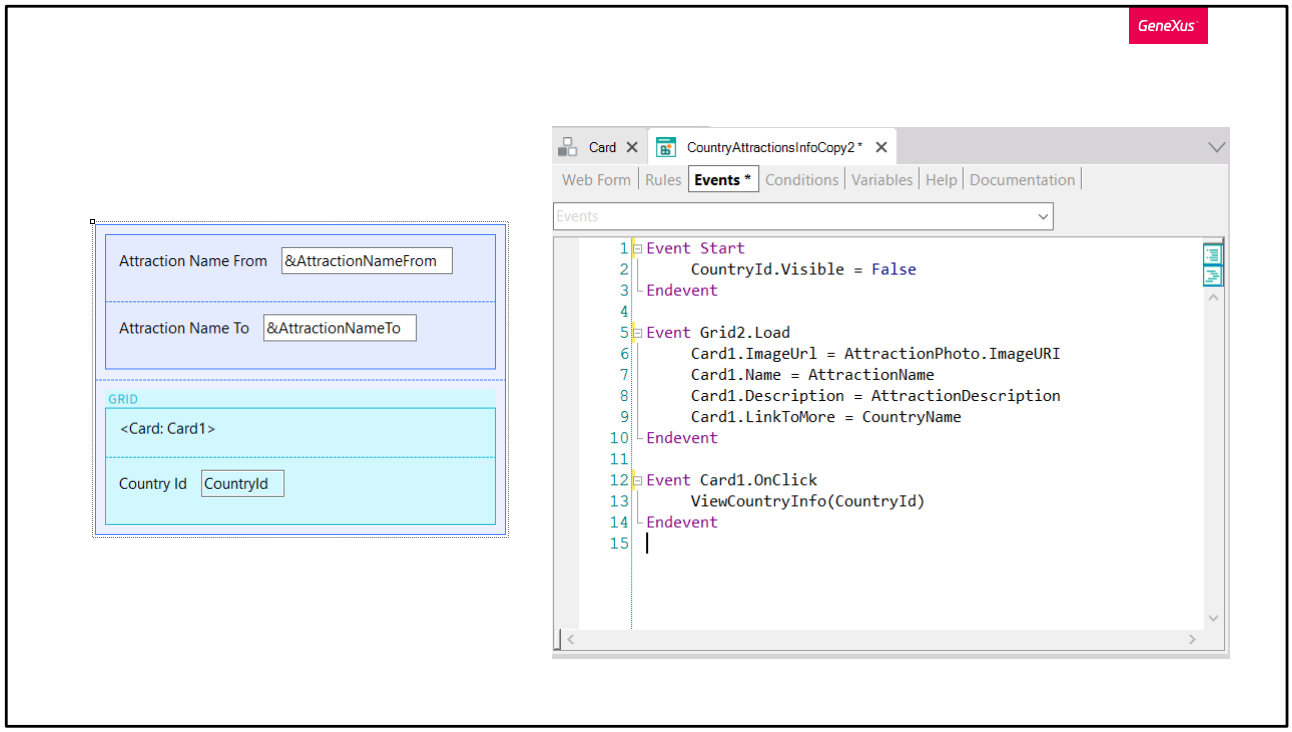
Lastly, we are going to have the Card User Control take its Base Style from there.

Well, it's as simple as that.



We already have the User Control to be used in our KB. What we are going to do now is save this component with another name and insert into the grid this Card User Control.

In its properties we can see displayed all those we changed, in a static manner to assign them a value, and –as we will see– in a dynamic manner too.

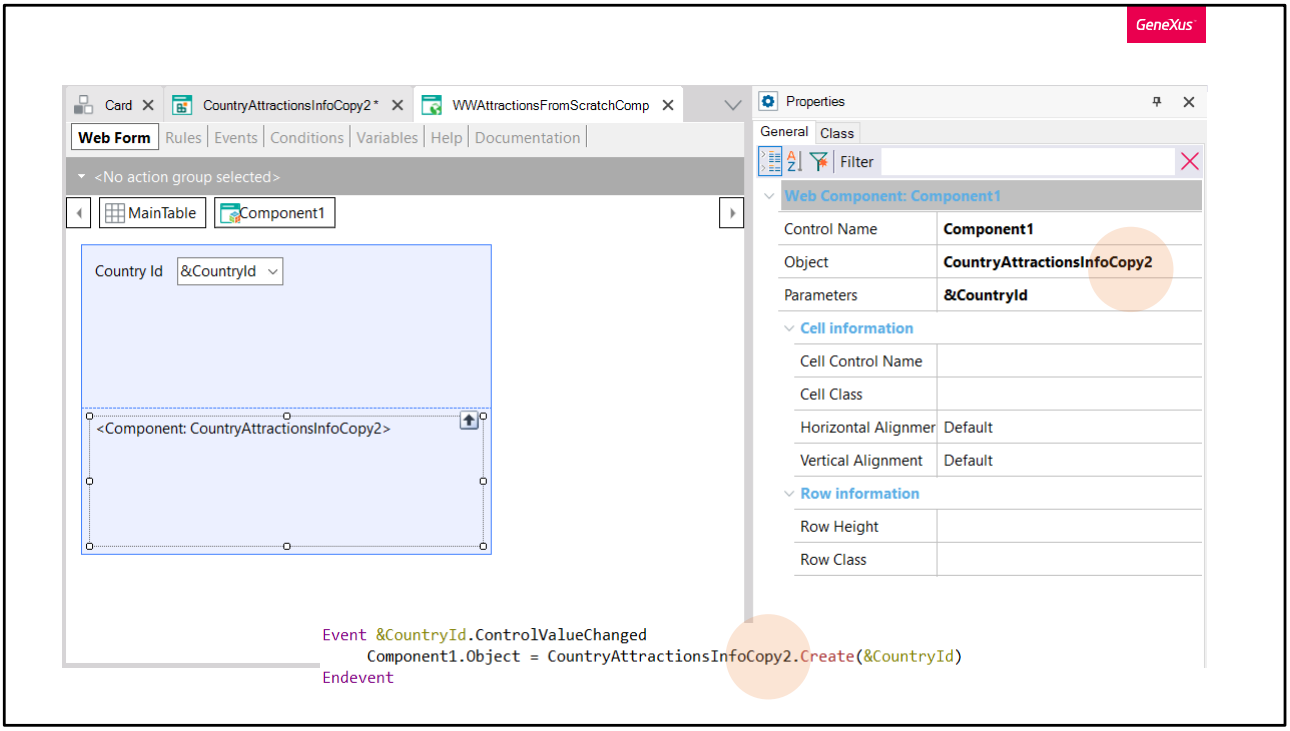


We are going to remove everything that we are no longer interested in. We are going to add the CountryId attribute, because we will want to send it as a parameter.
AttractionId no longer interests us. And we're going to set CountryId as invisible.

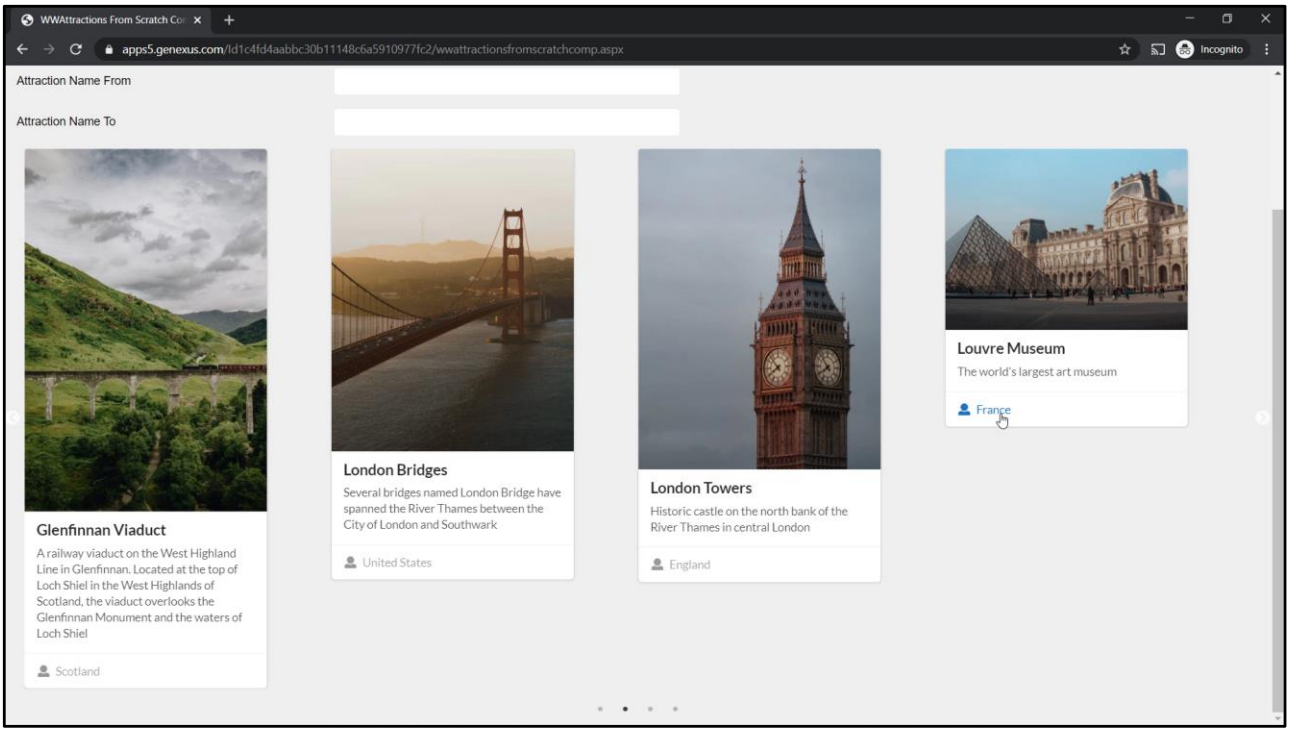
OK, what we have to do now is load that User Control to the Load event with the values of each attraction.
So, we program the Load event of the grid. And we see that by typing the name of the control, Card1, we already have all the properties that we saw before in the User Control.

To the Name property we are going to assign the value of the AttractionName attribute.
In the ImageUrl property we have to indicate the URL where the image we want to load in that Card will be found. And it is that of the AttractionPhoto attribute.
For the description, we see that we have added to the Attraction transaction the AttractionDescription attribute, which contains the description of the tourist attraction. We then load that element of the User Control with the attribute value.

Finally, this is going to be the link. We want the CountryName value to be loaded; that is to say, to indicate the name of the country. What we are going to do next is to create a link there, so that CountryName is a link to the panel we had defined to list the information of the country, and its attractions and cities. Therefore, we no longer need the event we had before, which was the one executed when clicking on AttractionName, and now we have to program the OnClick event of the user control. And what we will do is invoke the panel that we had, passing it the CountryId as a parameter.

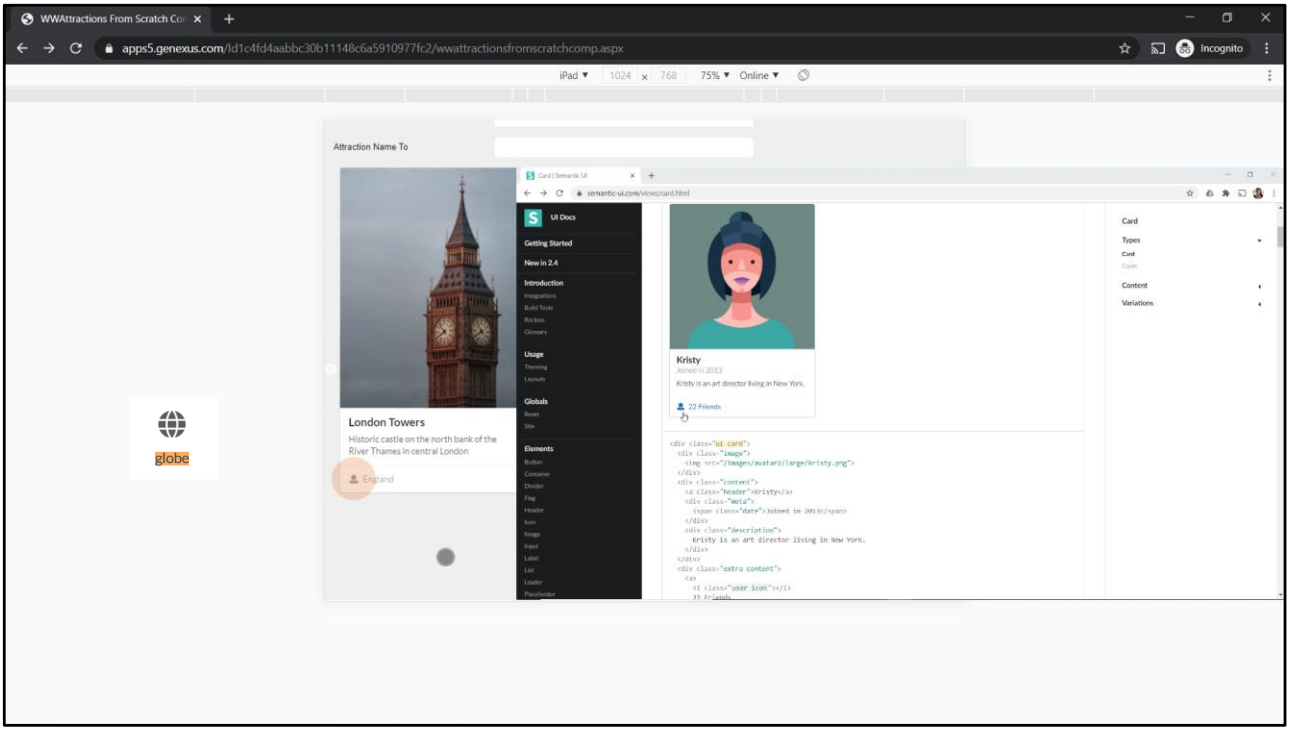


Finally, we will make some changes here. We want the component to be loaded with the web component we've just created.
Now we run it.

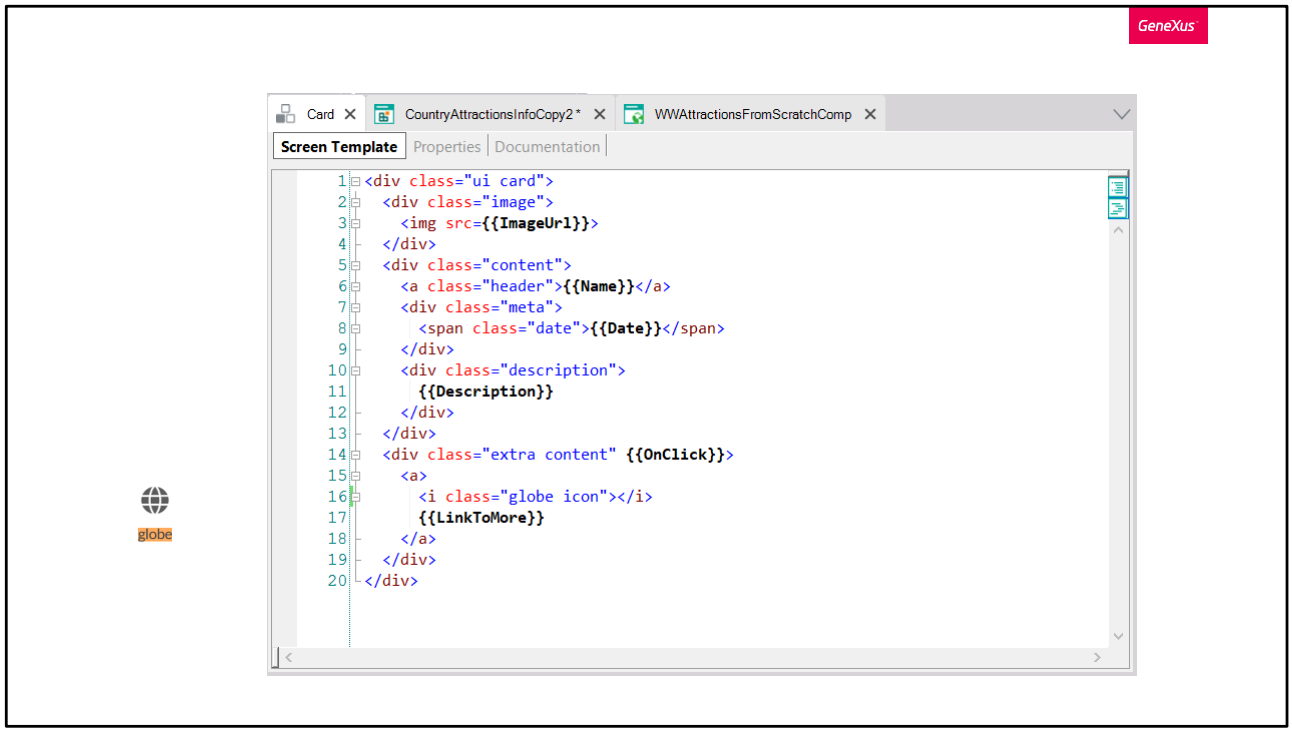


And see that the cards are displayed, as well as the view of the country.

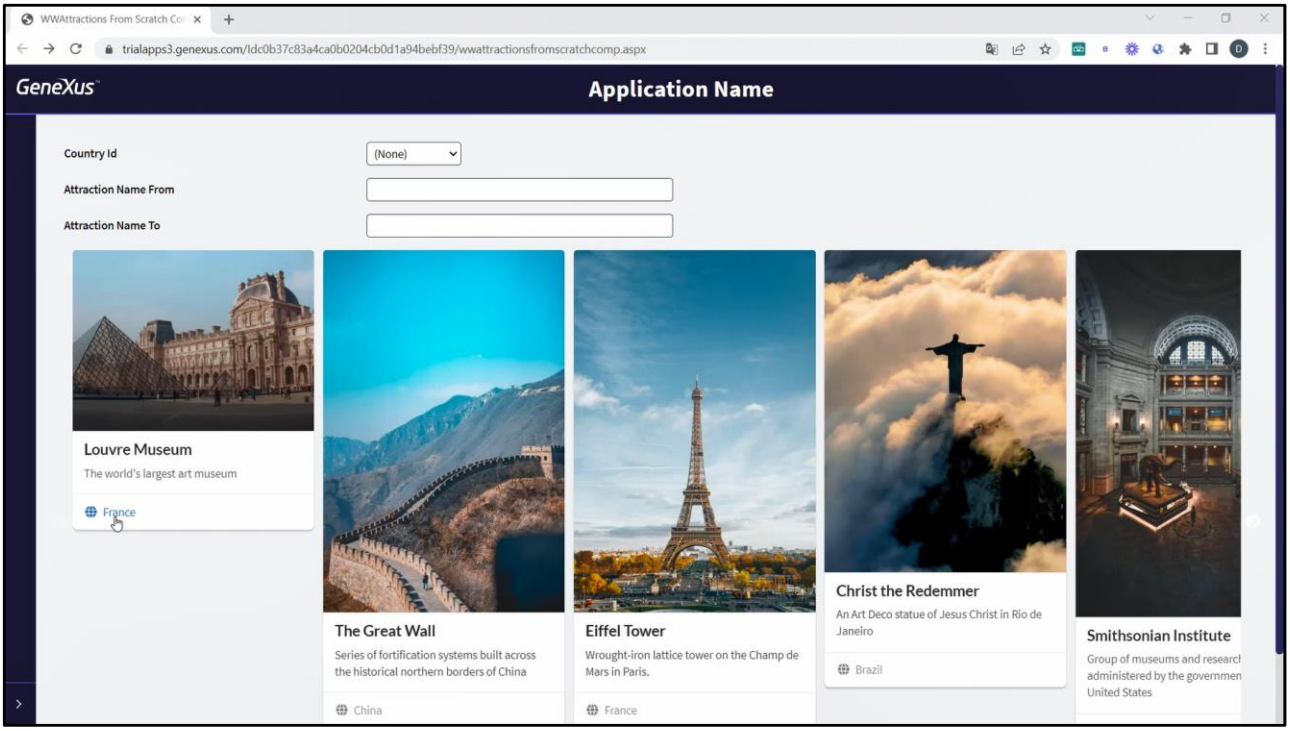
Here we are viewing 4 cards per page because we had set that for that size, for the Large size, the horizontal grid would show 4 columns, and if the size was Small it would show 2, and 3 when it was Medium.



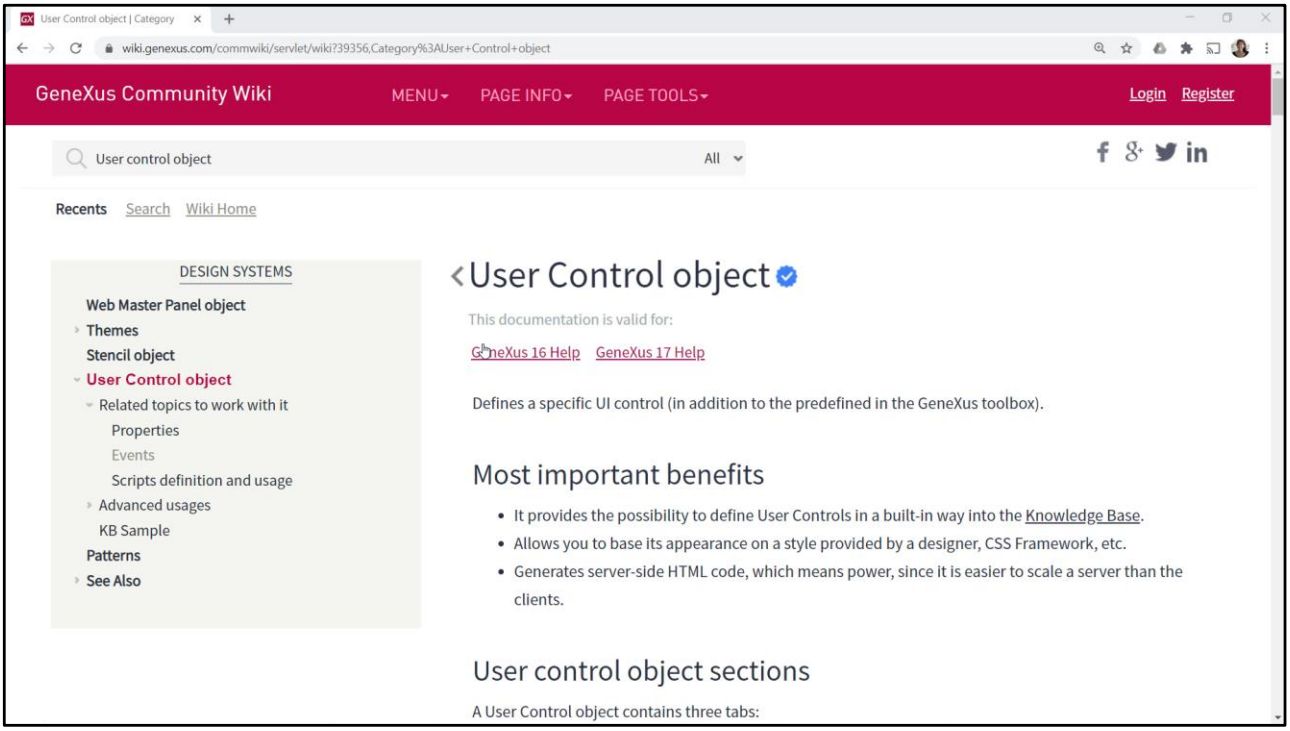
We see that the icon being displayed is a user icon that is not the one we are interested in We want an icon that reflects the reality of a country. So we are going to keep this one, the Globe icon.



The only thing we have to do here is change the class to Globe icon, because it is specified that way in Semantic UI.



Let's try and see how the icon is coming out.



This shows how easy it is to use an external User Control. In fact, we could create our own User Controls, writing in this way the HTML and providing the corresponding CSS library.

To learn more about User Controls, we recommend visiting our wiki.

GeneXus™

training.genexus.com
wiki.genexus.com